Model Predictive Control and Reinforcement Learning
– Policy Gradient and Actor-Critic Methods –

Joschka Boedecker and Moritz Diehl

University Freiburg

October 10, 2023

universität freiburg

Lecture Overview

1. Policy Gradient Methods
2. REINFORCE
3. Proximal Policy Optimization
4. Actor-Critic Methods
5. DDPG
6. TD3
7. Soft Actor-Critic

Acknowledgement

Slide contents are partially based on *Reinforcement Learning: An Introduction* by Sutton and Barto and the Reinforcement Learning lecture by David Silver.

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

Policy Gradient Methods

- Up to this point, we represented a model or a value function by some parameterized function approximator and extracted the policy implicitly
- Today, we are going to talk about *Policy Gradient Methods*: methods which consider a parameterized *policy*

$$\pi(a|s, \boldsymbol{\theta}) = \Pr\{A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\},$$

  with parameters $\boldsymbol{\theta}$
- Policy Gradient Methods are able to represent stochastic policies and scale naturally to very large or continuous action spaces

Policy Gradient Methods

- We update these parameters based on the gradient of some performance measure $J(\boldsymbol{\theta})$ that we want to maximize, i.e. via *gradient ascent*:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)},$$

where $\widehat{\nabla J(\boldsymbol{\theta}_t)} \in \mathbb{R}^d$ is a stochastic estimate whose expectation approximates the gradient of the performance measure w.r.t. $\boldsymbol{\theta}_t$

Policy Gradient Theorem

Policy Objective Functions:

- For episodic problems we define performance as: $J(\boldsymbol{\theta}) = \eta(\pi_{\boldsymbol{\theta}}) = \mathbb{E}_{s_0 \sim \rho_0}[v_{\pi_{\boldsymbol{\theta}}}(s_0)]$
- For continuing problems: $J(\boldsymbol{\theta}) = \sum_s \mu(s) v_{\pi_{\boldsymbol{\theta}}}(s)$

The Policy Gradient Theorem establishes that

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s, \boldsymbol{\theta})$$

Reminder: $v_{\pi_{\boldsymbol{\theta}}} = \sum_a \pi(a|s) q_\pi(s,a)$

Score Function

- Likelihood ratios exploit the following identity:

$$\overbrace{\nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})}^{\text{We want the expectation of this}} = \pi(a|s,\boldsymbol{\theta}) \frac{\nabla_{\boldsymbol{\theta}} \pi(a|s,\boldsymbol{\theta})}{\pi(a|s,\boldsymbol{\theta})}$$
$$= \underbrace{\pi(a|s,\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta})}_{\text{Easy to take the expectation because we can sample from } \pi!}$$

- $\nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta})$ is called the **score function**

Policy Gradient Theorem

Policy Gradient Theorem

For any differentiable policy $\pi(a|s,\boldsymbol{\theta})$ and any of the above policy objective functions, the policy gradient is:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[\nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta}) q_{\pi}(s,a)]$$

Score Function: Example

Consider a Gaussian policy, where the mean is a linear combination of state features: $\pi(a|s,\boldsymbol{\theta}) \sim \mathcal{N}(s^\top \boldsymbol{\theta}, \sigma^2)$, i.e.

$$\pi(a|s,\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2}\frac{(s^\top \boldsymbol{\theta} - a)^2}{\sigma^2})$$

Derivation of the score function

The log yields

$$\log \pi(a|s,\boldsymbol{\theta}) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(s^\top \boldsymbol{\theta} - a)^2$$

and the gradient

$$\nabla_{\boldsymbol{\theta}} \log \pi(a|s,\boldsymbol{\theta}) = -\frac{1}{2\sigma^2}(s^\top \boldsymbol{\theta} - a)2s = \frac{(a - s^\top \boldsymbol{\theta})s}{\sigma^2}.$$

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

REINFORCE

- REINFORCE: Monte Carlo Policy Gradient
- Builds upon Monte Carlo returns as an unbiased sample of q_π
- However, therefore REINFORCE can suffer from high variance

REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Variance Reduction with Baselines

- Vanilla REINFORCE provides *unbiased* estimates of the gradient $\nabla J(\theta)$, but it can suffer from high variance
- Goal: reduce variance while remaining unbiased
- Observation: we can generalize the policy gradient theorem by including an arbitrary *action-independent baseline* $b(s)$, i.e.

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla \pi(a|s) \\ &= \sum_s \mu(s) \left[\sum_a q_\pi(s,a) \nabla \pi(a|s) - b(s) \underbrace{\nabla \sum_a \pi(a|s)}_{=0} \right] \\ &= \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s)\end{aligned}$$

- Baselines can reduce the variance of gradient estimates significantly!

Variance Reduction with Baselines

- A constant value can be used as a baseline
- The state-value function can be used as a baseline

Question

Is the Q-function a valid baseline?

Question

Assume an approximation of the state-value function as a baseline. Is REINFORCE then biased?

REINFORCE with Baselines

Indeed, an estimate of the state value function, $\hat{v}(S_t, w)$, is a very reasonable choice for $b(s)$:

REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

Proximal Policy Optimization

- We collect data with $\pi_{\boldsymbol{\theta}_{\text{old}}}$
- And we want to optimize some objective to get a new policy $\pi_{\boldsymbol{\theta}}$
- We can write $\eta(\pi_{\boldsymbol{\theta}})$ in terms of $\pi_{\boldsymbol{\theta}_{\text{old}}}$:

$$\eta(\pi_{\boldsymbol{\theta}}) = \eta(\pi_{\boldsymbol{\theta}_{\text{old}}}) + \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{\infty} \gamma^t \mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_t, a_t)]$$

where the **advantage function** is defined as

$$\begin{aligned}\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a) &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}, s_{t+1}\sim p}[q_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a) - v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s)] \\ &= \mathbb{E}_{\pi_{\boldsymbol{\theta}}, s_{t+1}\sim p}[r(s,a) + \gamma v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s') - v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s)]\end{aligned}$$

- Advantage: how much better or worse is every action than average?

Proximal Policy Optimization

Proof:

$$
\begin{aligned}
&\mathbb{E}_{\pi_{\boldsymbol{\theta}}}[\sum_{t=0}^{\infty}\gamma^t \mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_t,a_t)] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}},s_{t+1}\sim p}[\sum_{t=0}^{\infty}\gamma^t (r(s_t,a_t)+\gamma v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_{t+1})-v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_t))] \\
&= \mathbb{E}_{\pi_{\boldsymbol{\theta}},s_{t+1}\sim p}[-v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_0)+\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)] \\
&= \mathbb{E}_{s_0\sim p_0}[-v_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s_0)]+\mathbb{E}_{\pi_{\boldsymbol{\theta}},s_{t+1}\sim p}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)] \\
&= -\eta(\pi_{\boldsymbol{\theta}_{\text{old}}})+\eta(\pi_{\boldsymbol{\theta}})
\end{aligned}
$$

Proximal Policy Optimization

- In PPO, we *ignore* the change in state distribution and optimize a **surrogate objective**:

$$\begin{aligned} J_{\text{old}}(\theta) &= \mathbb{E}_{s\sim\pi_{\boldsymbol{\theta}_{\text{old}}},a\sim\pi_{\boldsymbol{\theta}}}[\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a)] \\ &= \mathbb{E}_{(s,a)\sim\pi_{\boldsymbol{\theta}_{\text{old}}}}\left[\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{\text{old}}}}\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a)\right] \end{aligned}$$

- Improvement Theory: $\eta(\pi_{\boldsymbol{\theta}}) \geq J_{\text{old}}(\theta) - c \cdot \max_s \text{KL}[\pi_{\boldsymbol{\theta}_{\text{old}}}||\pi_{\boldsymbol{\theta}}]$
- If we keep the KL-divergence between our old and new policies small, optimizing the surrogate is close to optmizing $\eta(\pi_{\boldsymbol{\theta}})$!

Proximal Policy Optimization

- Clipped Surrogate Objective:

$$\mathbb{E}_{(s,a)\sim\pi_{\boldsymbol{\theta}_{\text{old}}}}\left[\min(\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{\text{old}}}}\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a), \text{clip}(\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{\text{old}}}}, 1-\epsilon, 1+\epsilon)\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a))\right]$$

- Adaptive Penalty Surrogate Objective:

$$\mathbb{E}_{(s,a)\sim\pi_{\boldsymbol{\theta}_{\text{old}}}}\left[\frac{\pi_{\boldsymbol{\theta}}}{\pi_{\boldsymbol{\theta}_{\text{old}}}}\mathcal{A}_{\pi_{\boldsymbol{\theta}_{\text{old}}}}(s,a) - \beta\text{KL}[\pi_{\boldsymbol{\theta}_{\text{old}}}||\pi_{\boldsymbol{\theta}}]\right]$$

for *iteration* $i = 1, 2, \ldots$ **do**
  Run policy for T timesteps of N trajectories
  Estimate advantage function at all timesteps
  Do SGD on one of the above objectives for some number of epochs
  In case of the Adaptive Penalty Surrogate: Increase β if KL-divergence too high, otherwise decrease β
end

Algorithm 1: PPO

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

Actor-Critic Methods

- Methods that learn approximations to both policy and value functions (and use the critic for bootstrapping) are called actor-critic methods
  actor: learned policy
  critic: learned value function (usually a state-value function)

Question: Is REINFORCE-with-baseline considered as an actor-critic method?

Actor-Critic Methods

- REINFORCE-with-baseline is unbiased, but tends to learn slowly and has high variance
- To gain from advantages of TD methods we use actor-critic methods with a bootstrapping critic

One-step actor-critic methods

Replace the full return of REINFORCE with one-step return as follows:

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha \left(G_{t:t+1} - \hat{v}(S_t, \boldsymbol{w})\right) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}) - \hat{v}(S_t, \boldsymbol{w})\right) \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)} \\ &= \boldsymbol{\theta}_t + \alpha \delta_t \frac{\nabla \pi(A_t|S_t, \boldsymbol{\theta}_t)}{\pi(A_t|S_t, \boldsymbol{\theta}_t)}\end{aligned}$$

Actor-Critic Methods

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

Deep Deterministic Policy Gradient

- DDPG is an actor-critic method (*Continuous DQN*)
- Recall the DQN-target: $y_j = r_j + \gamma \max_a Q(s_{j+1}, a, \mathbf{w}^-)$
- In case of continuous actions, the maximization step is not trivial
- Therefore, we approximate deterministic actor μ representing the $\arg\max_a Q(s_{j+1}, a, \mathbf{w})$ by a neural network and update its parameters following the *Deterministic Policy Gradient Theorem*:

$$\nabla_\theta \leftarrow \frac{1}{N} \sum_j \nabla_a Q(s_j, a, \mathbf{w})|_{a=\mu(s_j)} \nabla_\theta \mu(s_j, \boldsymbol{\theta})$$

- Exploration by adding Gaussian noise to the output of μ

Deep Deterministic Policy Gradient

- The Q-function is fitted to the adapted TD-target:

$$y_j = r_j + \gamma Q(s_{j+1}, \mu(s_{j+1}, \boldsymbol{\theta}^-), \mathbf{w}^-)$$

- The parameters of target networks $\mu(\cdot, \boldsymbol{\theta}^-)$ and $Q(\cdot, \cdot, \mathbf{w}^-)$ are then adjusted with a soft update

$$\mathbf{w}^- \leftarrow (1-\tau)\mathbf{w}^- + \tau \mathbf{w} \text{ and } \boldsymbol{\theta}^- \leftarrow (1-\tau)\boldsymbol{\theta}^- + \tau \boldsymbol{\theta}$$

  with $\tau \in [0, 1]$
- DDPG is very popular and builds the basis for more SOTA actor-critic algorithms
- However, it can be quite unstable and sensitive to its hyperparameters

Deep Deterministic Policy Gradient

Initialize replay memory D to capacity N
Initialize critic Q and actor μ with random weights
for *episode* $i = 1, .., M$ **do**
  for $t = 1, .., T$ **do**
    select action $a_t = \mu(s_t, \boldsymbol{\theta}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$
    Store transition (s_t, a_t, s_{t+1}, r_t) in D
    Sample minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
    Set $y_j = \begin{cases} r_j & \text{if } s_{j+1} \text{ is terminal} \\ r_j + \gamma\, Q(s_{j+1}, \mu(s_{j+1}, \boldsymbol{\theta}^-), \mathbf{w}^-) & \text{else} \end{cases}$
    Update the parameters of Q according to the TD-error
    Update the parameters of μ according to:

$$\nabla_\theta \leftarrow \frac{1}{N} \sum_j \nabla_a Q(s_j, a, \mathbf{w})|_{a=\mu(s_j)} \nabla_\theta \mu(s_j, \boldsymbol{\theta})$$

    Adjust the parameters of the target networks via a soft update
  end
end

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

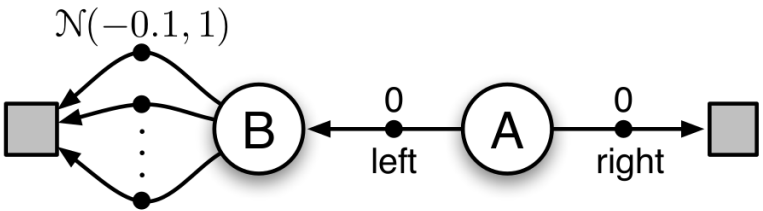
5 DDPG

6 TD3

7 Soft Actor-Critic

Overestimation Bias

- In all control algorithms so far, the target policy is created by the *maximization* of a value-function
- We thus consider the maximum over estimated values as an estimate of the maximum value
- This can lead to the so-called *overestimation bias*



Overestimation Bias

- Recall the Q-learning target: $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- Imagine two random variables X_1 and X_2:

$$\mathbb{E}[\max(X_1, X_2)] \geq \max(\mathbb{E}[X_1], \mathbb{E}[X_2])$$

- $Q(S_{t+1}, a)$ is not perfect – it can be *noisy*:

$$\max_a Q(S_{t+1}, a) = \overbrace{Q(S_{t+1}, \underbrace{\arg\max_a Q(S_{t+1}, a)}_{\text{action comes from } Q})}^{\text{value comes from } Q}$$

- If the noise in these is decorrelated, the problem goes away!

Double Q-learning

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

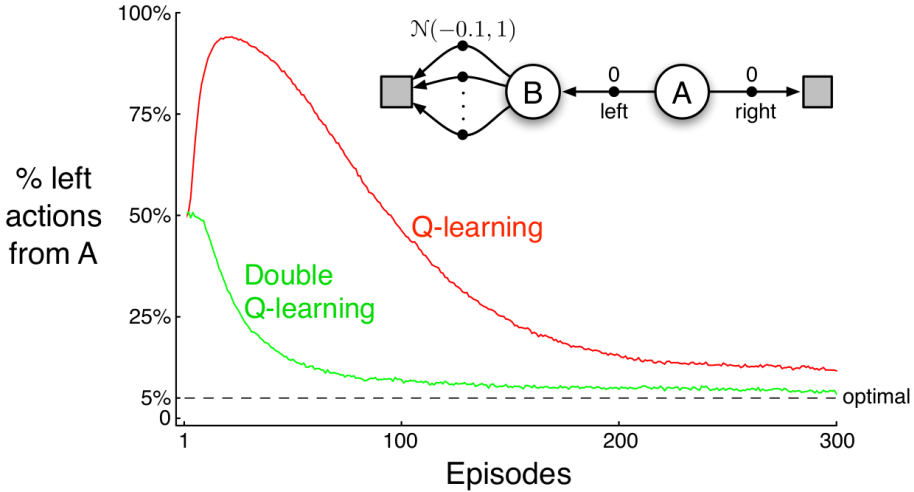
$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \operatorname{argmax}_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \operatorname{argmax}_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

Double Q-learning



TD3

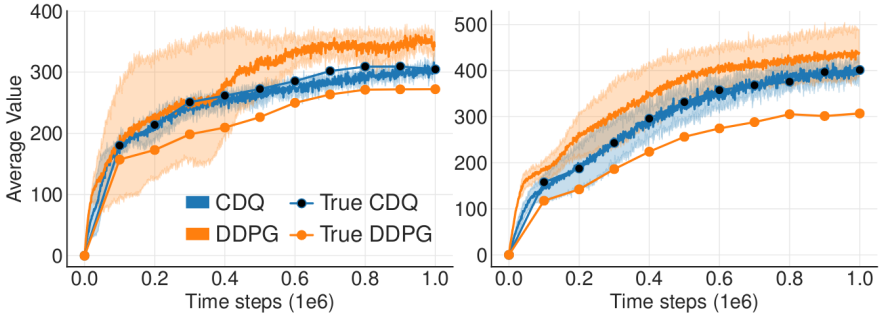
TD3 adds three adjustments to vanilla DDPG

- Clipped Double Q-Learning
- Delayed Policy Updates
- Target-policy smoothing

TD3: Clipped Double Q-Learning

- In order to alleviate the overestimation bias (which is also present in actor-critic methods), TD3 learns two approximations of the action-value function
- It the takes the minimum of both predictions as the second part of the TD-target:

$$y_j = r_j + \gamma \min_{i \in \{1,2\}} Q(s_{j+1}, \mu(s_{j+1}), \mathbf{w}_i^-)$$



TD3: Delayed Policy Updates

- Due to the mutual dependency between actor and critic updates...
  - values can diverge when the policy leads to overestimation and
  - the policy will lead to bad regions of the state-action space when the value estimates lack in (relative) accuracy
- Therefore, policy updates on states where the value-function has a high prediction error can cause divergent behaviour
- We already know how to compensate for that: target networks
- Freeze target and policy networks between d updates of the value function
- This is called a *Delayed Policy Update*

TD3: Target-policy Smoothing

- *Target-policy Smoothing* adds Gaussian noise to the next action in target calculation
- It transforms the Q-update towards an Expected SARSA update fitting the value of a small area around the target-action:

$$y_j = r_j + \gamma \min_{i \in \{1,2\}} Q(s_{j+1}, \mu(s_{j+1}) + \mathsf{clip}(\epsilon, -c, c), \mathbf{w}_i^-),$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$

TD3: Ablation

Table 2. Average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over delayed policy updates (DP), target policy smoothing (TPS), Clipped Double Q-learning (CDQ) and our architecture, hyper-parameters and exploration (AHE). Maximum value for each task is bolded.

Method	HCheetah	Hopper	Walker2d	Ant
TD3	9532.99	**3304.75**	**4565.24**	**4185.06**
DDPG	3162.50	1731.94	1520.90	816.35
AHE	8401.02	1061.77	2362.13	564.07
AHE + DP	7588.64	1465.11	2459.53	896.13
AHE + TPS	9023.40	907.56	2961.36	872.17
AHE + CDQ	6470.20	1134.14	3979.21	3818.71
TD3 - DP	9590.65	2407.42	**4695.50**	3754.26
TD3 - TPS	8987.69	2392.59	4033.67	**4155.24**
TD3 - CDQ	9792.80	1837.32	2579.39	849.75
DQ-AC	9433.87	1773.71	3100.45	2445.97
DDQN-AC	**10306.90**	2155.75	3116.81	1092.18

Lecture Overview

1 Policy Gradient Methods

2 REINFORCE

3 Proximal Policy Optimization

4 Actor-Critic Methods

5 DDPG

6 TD3

7 Soft Actor-Critic

Soft Actor-Critic

- Soft Actor-Critic: entropy-regularized value-learning
- The policy is trained to maximize a trade-off between expected return and entropy ($H(P) = \mathbb{E}_{x\sim P}[-\log P(x)]$), a measure of randomness in the policy:

$$\pi_* = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} R_{t+1} + \alpha H(\pi(\cdot|S_t = s_t)) \right]$$

- The value functions are then defined as:

$$v_\pi(s) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} R_{t+1} + \alpha H(\pi(\cdot|S_t = s_t))|S_0 = s, A_0 = a]$$

and

$$q_\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} R_{t+1} + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot|S_t = s_t))|S_0 = s, A_0 = a]$$

- And their relation as: $v_\pi(s) = \mathbb{E}_\pi[q_\pi(s,a)] + \alpha H(\pi(\cdot|S_t = s))$

Soft Actor-Critic

- The corresponding Bellman equation for q_π is

$$\begin{aligned} q_\pi(s_t, a_t) &= \mathbb{E}_{\pi,p}[R_{t+1} + \gamma(q_\pi(s_{t+1}, a_{t+1}) + \alpha H(\pi(\cdot|S_{t+1} = s_{t+1}))] \\ &= \mathbb{E}_{\pi,p}[R_{t+1} + \gamma v_\pi(s_{t+1})] \end{aligned}$$

- Loss for the Q-networks:

$$L\left(\mathbf{w}_i, \mathcal{D}\right) = \mathop{\mathrm{E}}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q(s, a, \mathbf{w}_i) - y\left(r, s'\right)\right)^2\right]$$

where the target is:

$$y\left(r, s'\right) = r + \gamma\left(\min_{j=1,2} Q\left(s', \tilde{a}', \mathbf{w}_j^-\right) - \alpha \log \pi_\theta\left(\tilde{a}' \mid s'\right)\right), \quad \tilde{a}' \sim \pi_\theta\left(\cdot \mid s'\right)$$

Soft Actor-Critic

- We want to find a policy which maximizes expected future return and expected future entropy, i.e. which maximizes $V^\pi(s)$:

$$\begin{aligned} V^\pi(s) &= \mathrm{E}_\pi \left[Q^\pi(s,a) \right] + \alpha H(\pi(\cdot \mid s)) \\ &= \mathrm{E}_\pi \left[Q^\pi(s,a) - \alpha \log \pi(a \mid s) \right] \end{aligned}$$

- To optimize the policy despite the sampling of actions, we make use of the *reparameterization trick*:

$$\tilde{a}_\theta(s,\xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \quad \xi \sim \mathcal{N}(0,I)$$

- We can thus rewrite the expectation from above as:

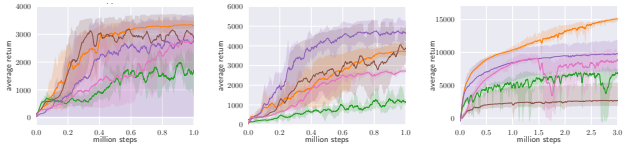
$$\mathrm{E}_{\pi_\theta} \left[Q^{\pi_\theta}(s,a) - \alpha \log \pi_\theta(a \mid s) \right] = \mathrm{E}_\xi \left[Q^{\pi_\theta}\left(s, \tilde{a}_\theta(s,\xi)\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s,\xi) \mid s\right) \right]$$

Soft Actor-Critic

- Final policy loss is then:

$$\max_{\theta} \mathrm{E}_{s,\xi}\left[\min_{j=1,2} Q\left(s, \tilde{a}_\theta(s,\xi), \mathbf{w}_j^-\right) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s,\xi) \mid s\right)\right]$$

- Performance comparison from (Haarnoja et al., 2018):



(a) Hopper-v1 (b) Walker2d-v1 (c) HalfCheetah-v1

(d) Ant-v1 (e) Humanoid-v1 (f) Humanoid (rllab)

Further ressources

If you want to get an even more detailed overview about the current SOTA, you can have a look at OpenAI SpinningUp:

`https://spinningup.openai.com/en/latest/index.html`